浙江大学
ZHEJIANG UNIVERSITY

Computer Architecture Experiment

Topic 4. Pipelined CPU resolving control hazard

浙江大学计算机学院
陈文智
chenwz@zju.edu.cn

实验操作流程

- 阅读实验文档，理解处理器中的控制竞争问题的多种解决方案
- 以前一次实验为基础，通过predict-not-taken和delay-slot的方法解决控制竞争
- 对处理器进行仿真，检验处理器的仿真结果是否符合要求。
- 综合工程并下载至开发板，在单步执行的过程中检查调试屏幕的输出，检验处理器的执行过程是否正确。

实验验收标准

- 仿真执行过程中，处理器的行为和内部控制信号均符合要求。
- 下载至开发板后的单步执行过程中，寄存器的变化过程和最终执行结果与测试程序相吻合。

Outline

- Experiment Purpose
- Experiment Task
- Basic Principle
- Operating Procedures
- Precaution
- Checkpoints

Experiment Purpose

- **Understand the reason why and when Control Hazards arise**
- **Master the methods of resolving Control Hazards**
  - Freeze or flush
  - Predict-not-taken
  - Predict-taken
  - Delayed-branch
- **Master the methods of 1-cycle stall of Predict-not-taken and Predict-taken branch.**
- **Master the Condition In Which Bypass Unit doesn't Work and the Pipeline stalls.**
- **master methods of program verification of Pipelined CPU resolving control hazards**

Experiment Task

- **Improve the design of Datapath of 5-stages Pipelined CPU to implement 1-cycle stall when CPU takes Predict-taken policy.**
  - Bring forward calculation of condition & branch address
  - Bring forward bypass unit
- **Verify the Pipeline CPU with program and observe the execution of program**

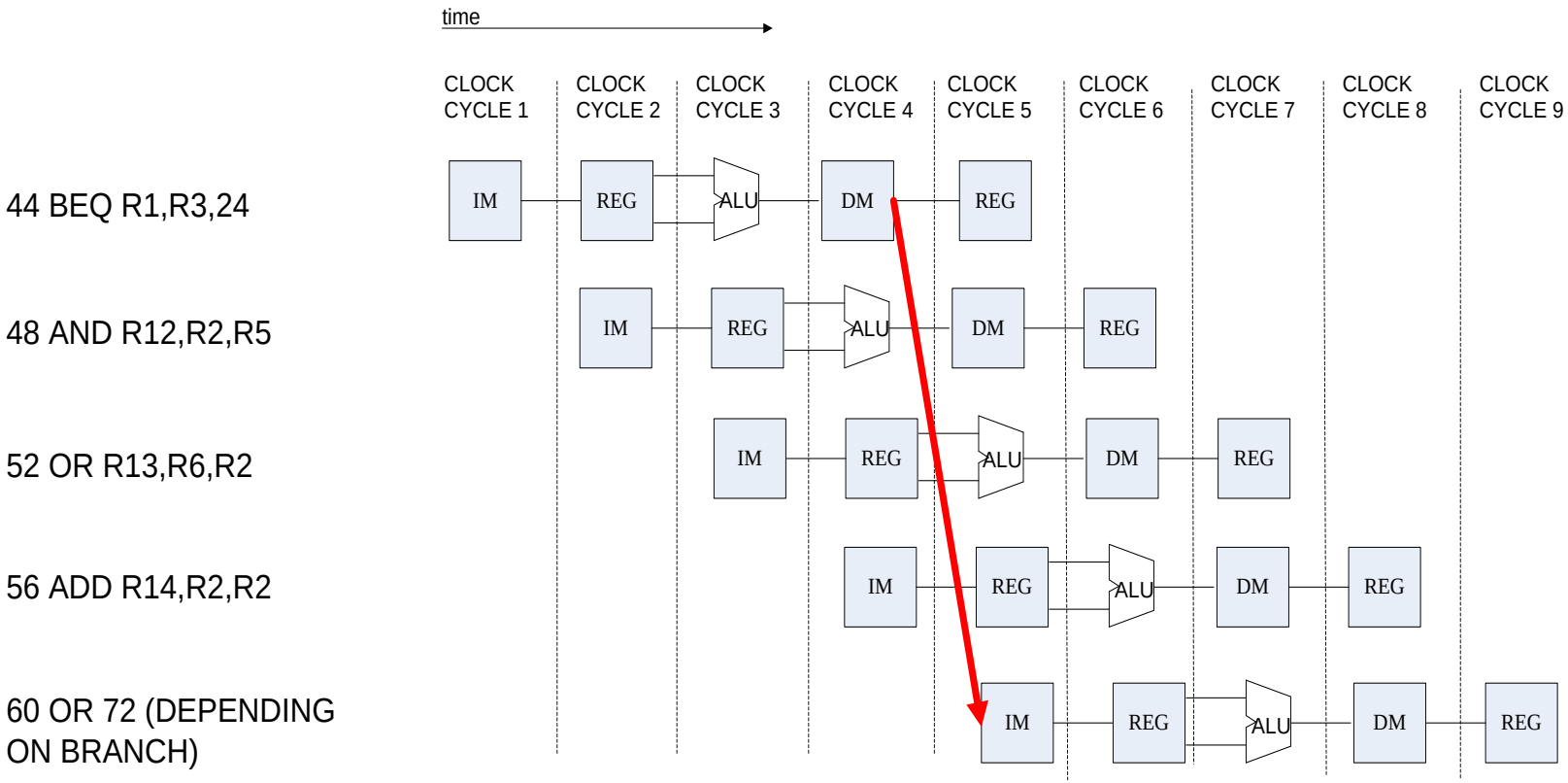
Control Hazard Definition

- **Control Hazards arise from the pipelining of branches and other instructions that change the PC.**
- **Control hazards can cause a greater performance loss for our MIPS pipeline than do data hazards.**
- **Reducing Pipeline Branch Penalties.**

Instruction Demo

	Address	Instruction
	36	Nop
	40	Add R30,R30,R30
Branch to 72→	44	Beq R1,R3,24
if R1!=R3, it executes in sequence	48	And R12,R2,R5
	52	Or R13,R6,R2
	56	And R14,R2,R2
	60	
	64	
	68	
If R1=R3, it executes these instruction	72	Lw R14,50(R7)
	76	

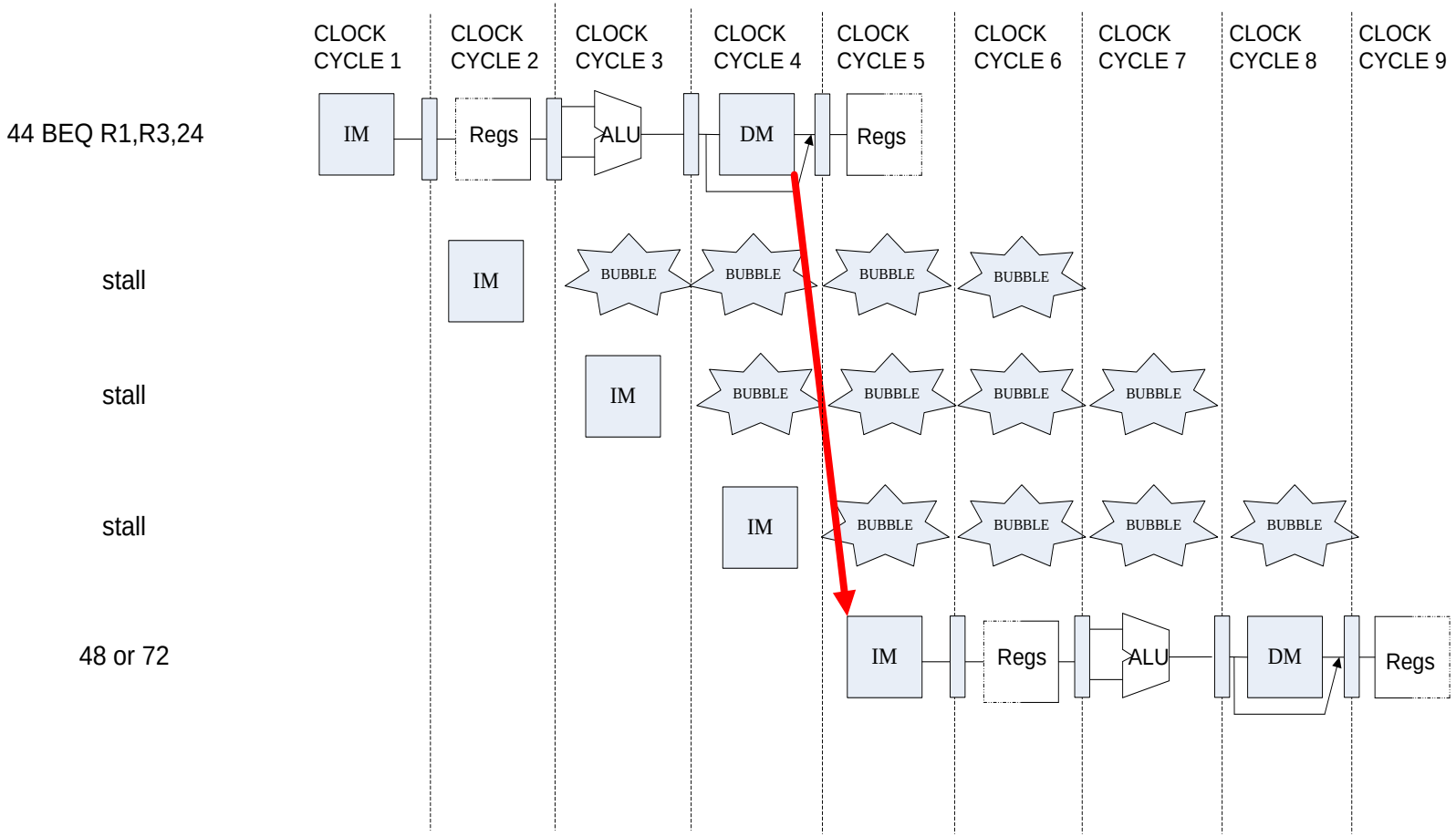
Execution result



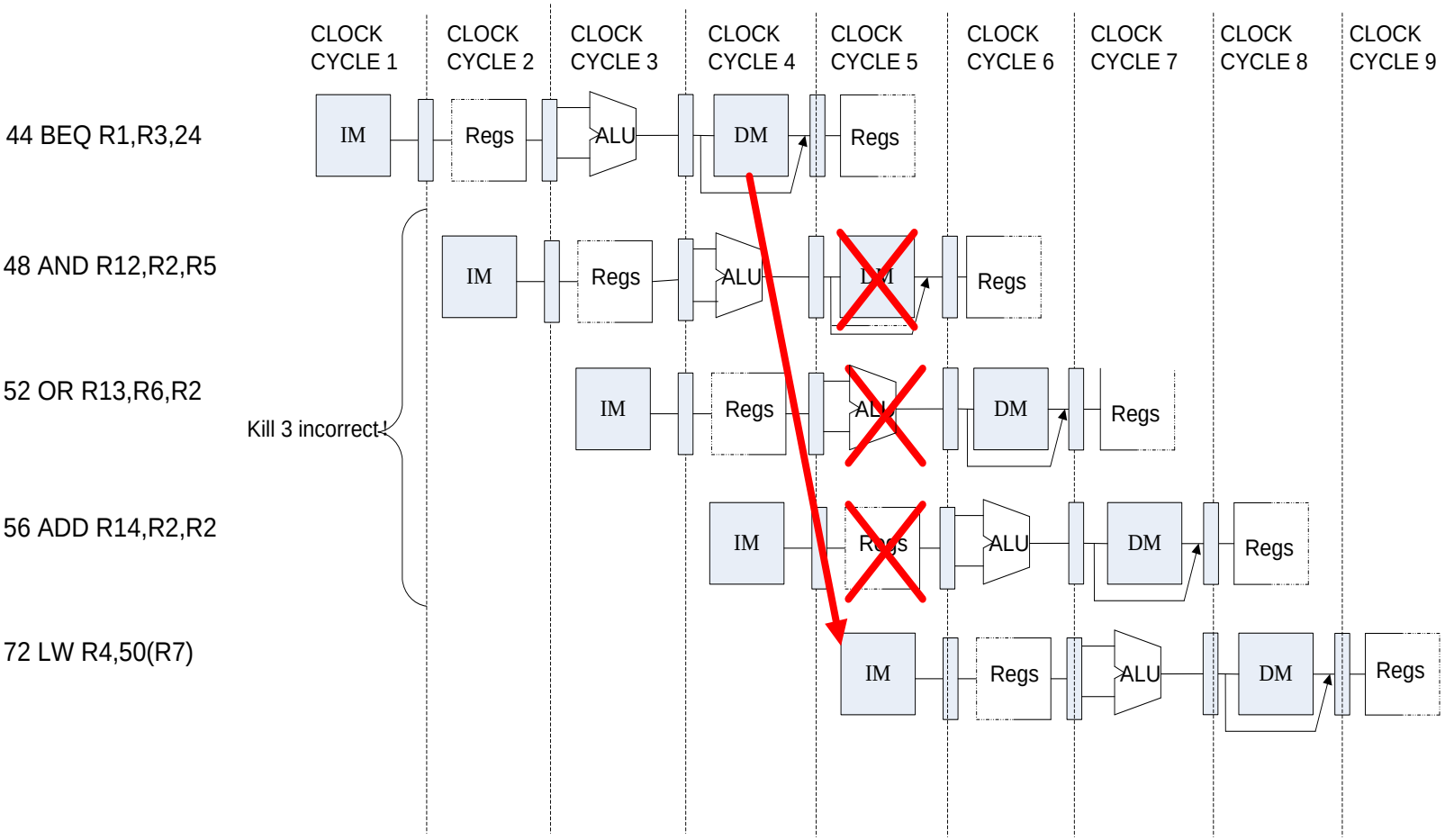
Methods of resolving Control hazards

- **Freeze or flush the pipeline**
- **Predict-not-taken**
- **Predict-taken**
- **Delayed branch**

Freeze method



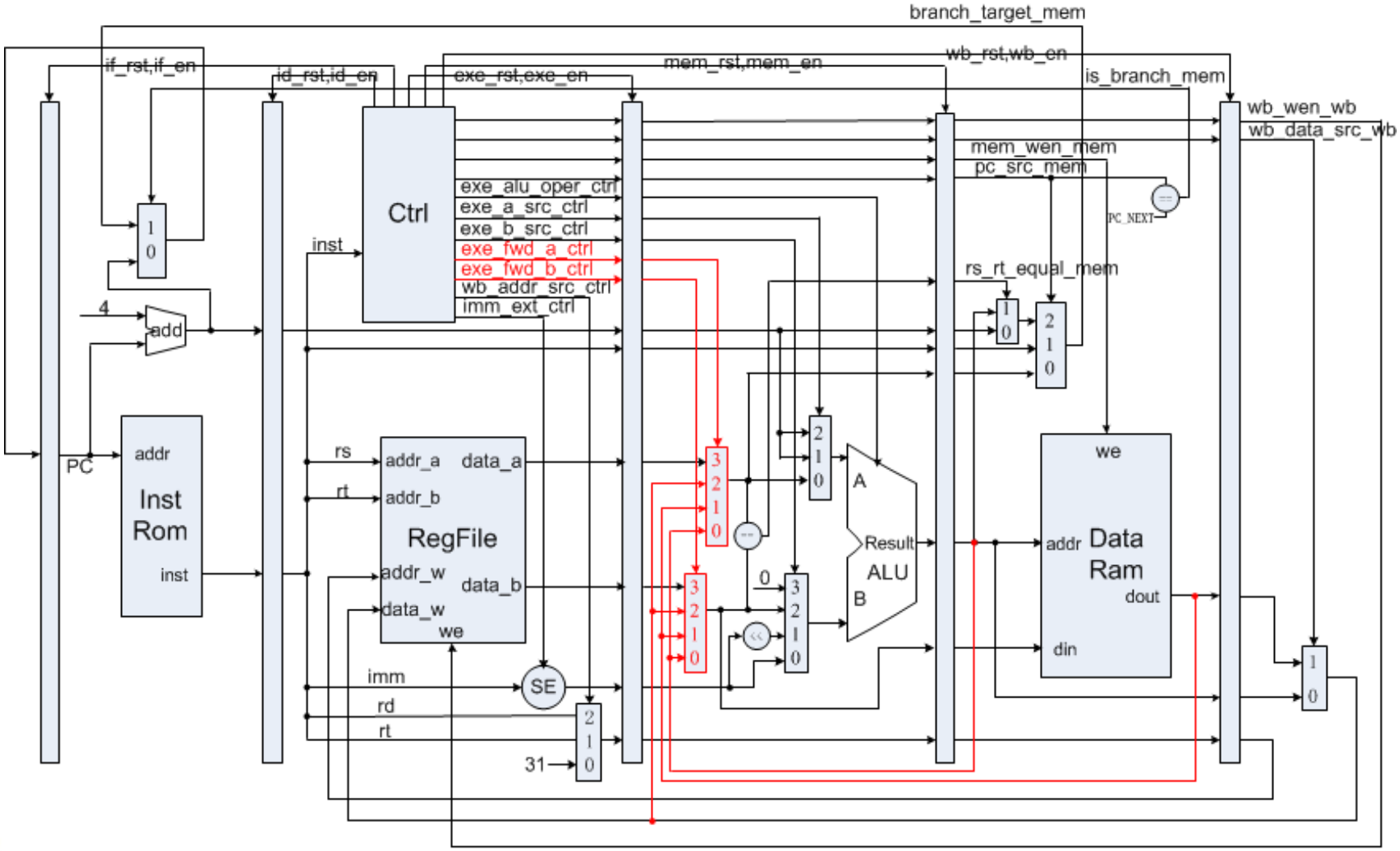
Predict-not-taken



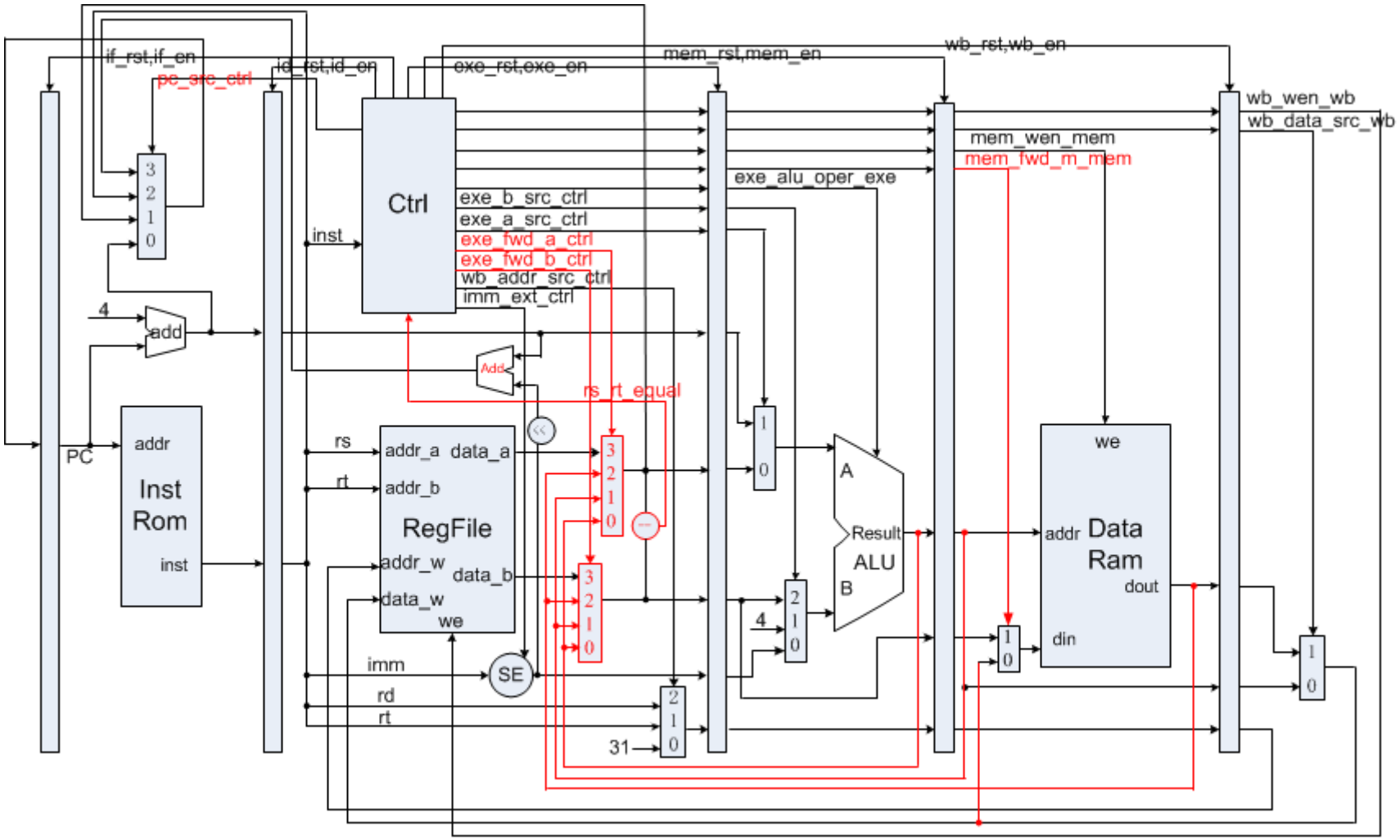
Predict-taken method

- **60% of branch result is taken**
- **Bring forward calculation of branch condition from MEM Stage to EX Stage, stall reduce from 3-cycle to 2-cycle.**
- **Then bring forward from EX to ID, stall reduce from 2-cycle to 1-cycle.**
- **1-cycle stall may be used for 1 delay slot**

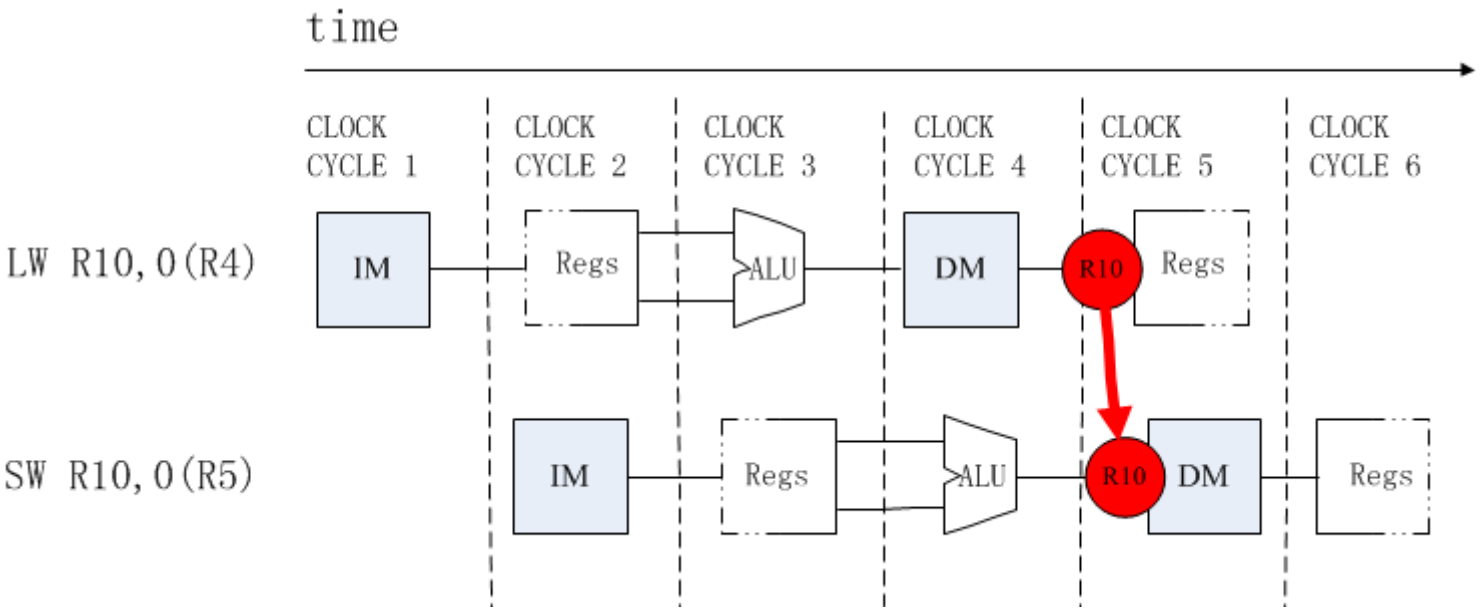
Original Datapath



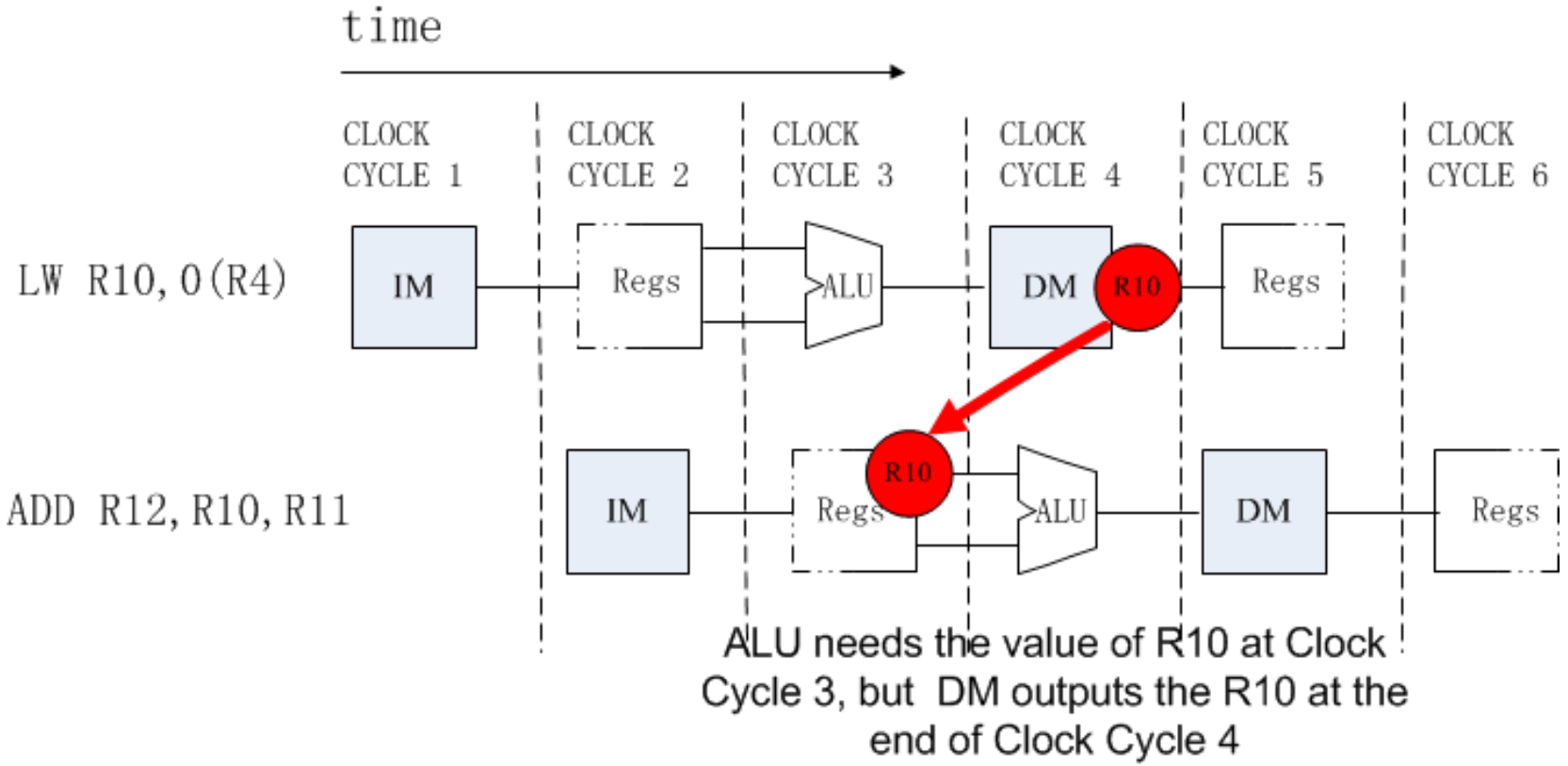
Datapath resolving Control Hazards



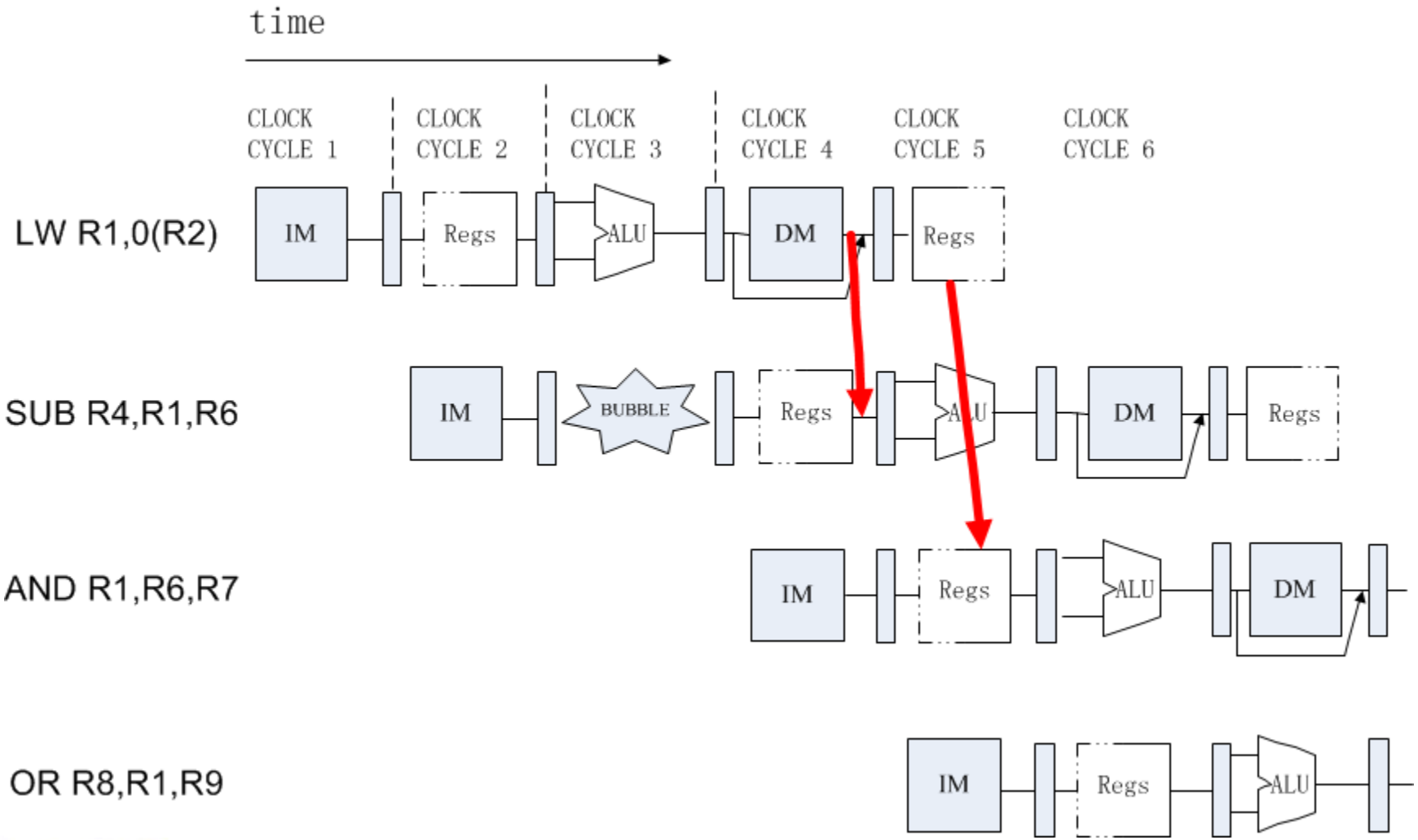
SW After LW/R-Type



Condition in Which Bypass Unit doesn't work



Pipeline stalls at ID Stage



load_stall &&fwd_m in Controller

- **load_stall = 1**
  - rs_used&& regw_addr_exe == addr_rs &&
    wb_wen_exe && is_load_exe
  - rt_used&&regw_addr_exe == addr_rt &&
    wb_wen_exe&&is_load_exe&&~is_store
- **fwd_m = 1**
  - rt_used&&regw_addr_exe == addr_rt &&
    wb_wen_exe&&is_load_exe&&is_store

Program for verification (1)

```
00000824 main:    and $1, $0, $0      # address of data[0]
34240050          ori $4, $1, 80      # address of data[0]
0c00000a call:    jal sum             # call function
20050004          addi $5, $0, 4      # counter
ac820000 return:  sw $2, 0($4)        # store result
8c890000          lw $9, 0($4)        # check sw
ac890004          sw $9, 4($4)        # store result again
01244022          sub $8, $9, $4      # sub: $8 <- $9 - $4
08000008 finish:  j finish            # dead loop
00000000          nop                 # done
```

Program for verification (2)

```
00004020 sum:   add $8, $0, $0      # sum function entry
8c890000 loop:  lw $9, 0($4)        # load data
01094020        add $8, $8, $9      # sum
20a5ffff        addi $5, $5, -1     # counter - 1
0005182a        slt $3, $0, $5      # finish?
1460fffb        bne $3, $0, loop    # finish?
20840004        addi $4, $4, 4      # address + 4
03e00008        jr $ra              # return
01001025        or $2, $8, $0       # move result to $v0
00000000        nop                 # done
```

Checkpoints

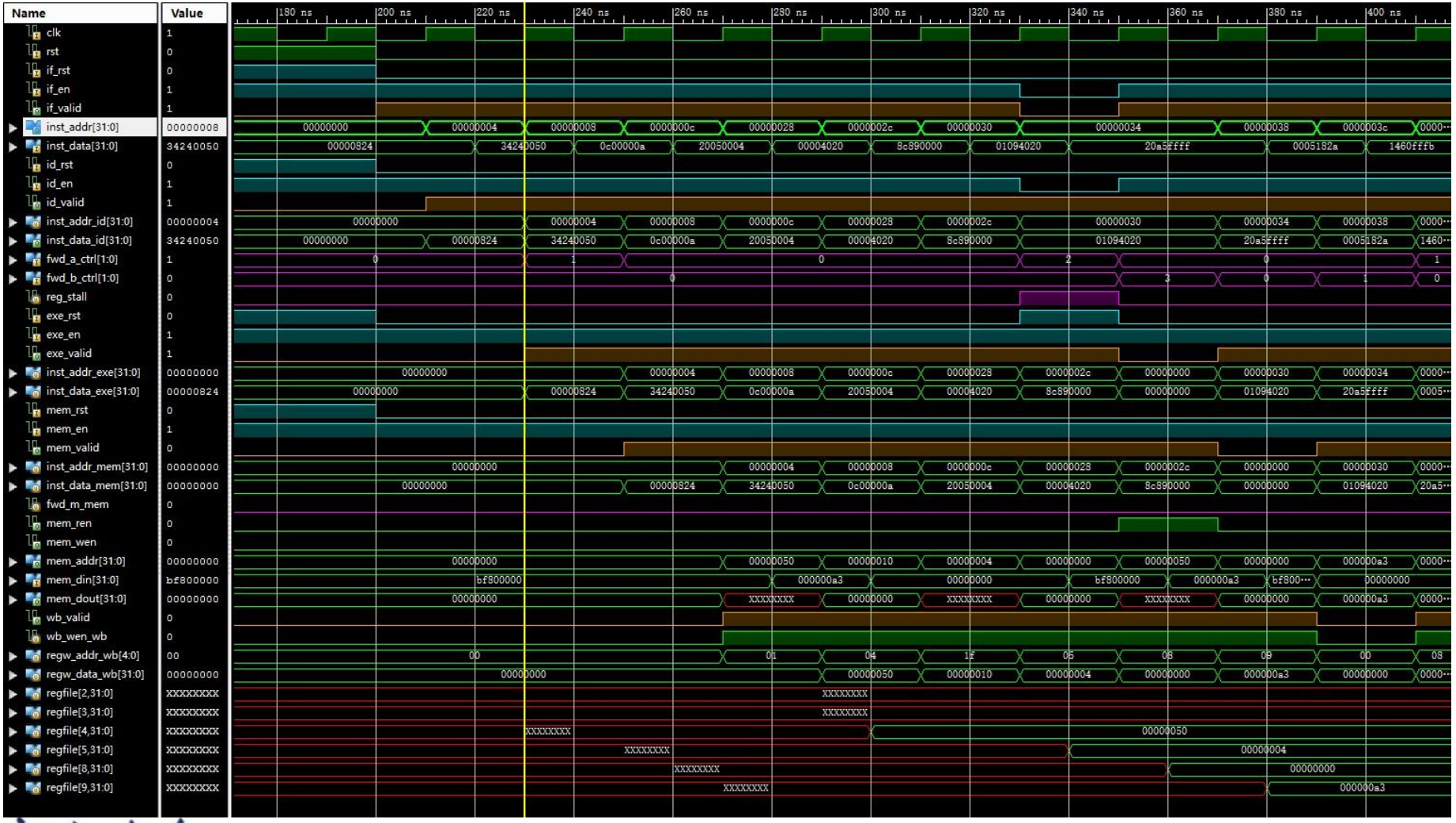
- **CP 1:**

  Waveform Simulation of the Pipelined CPU with the verification program

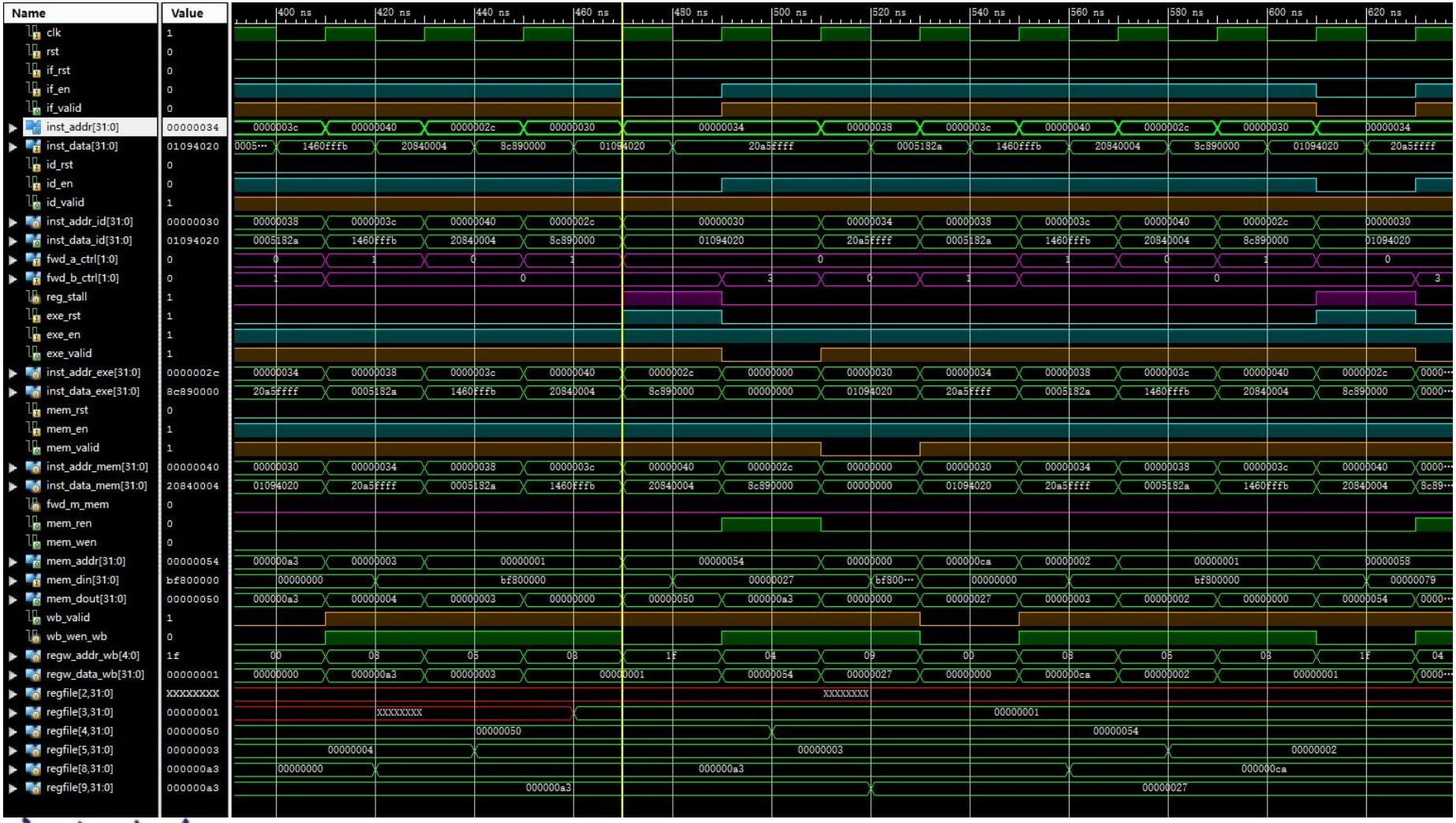
- **CP 2:**

  FPGA Implementation of the Pipelined CPU with the verification program

Simulation (1)

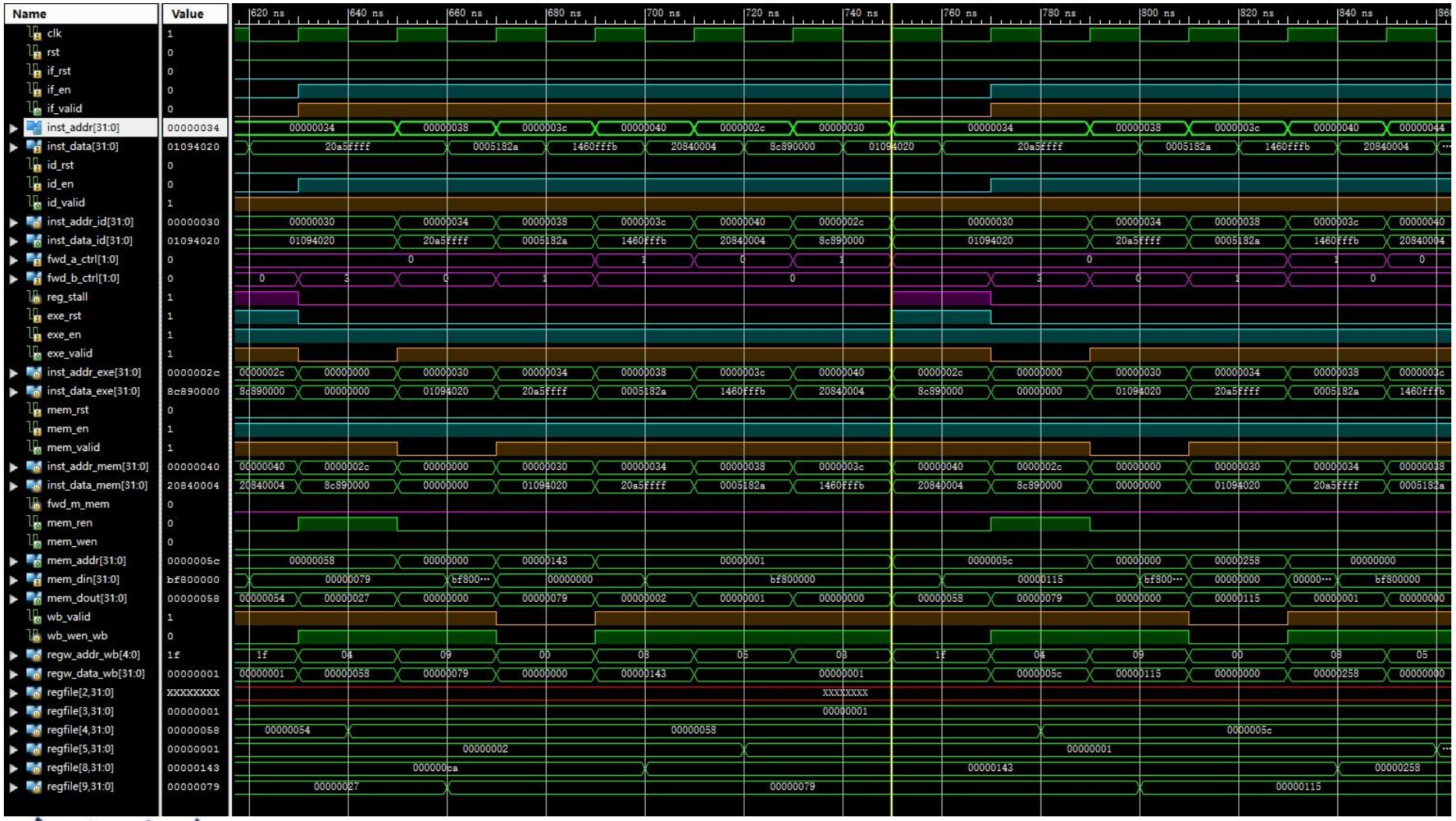


Simulation (2)



浙江大学 ZheJiang University

Simulation (3)



ZheJiang University

Thanks!

浙江大学
ZheJiang University